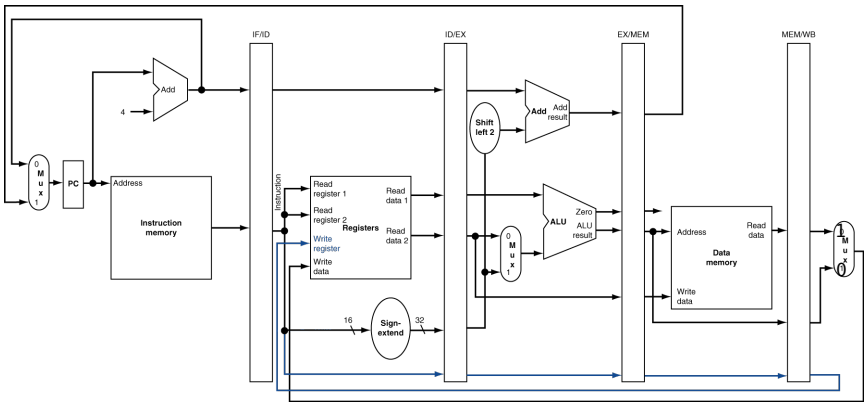
Each instruction type has own operations after ID:

Stage	lw	sw	ALU	beq
EX	ADDR = rs + I	ADDR = rs + I	Result = rs op rt	Compute BTA; rs - rt
MEM	Data = MEM[ADDR]	MEM[ADDR] = rt		If (rs-rt == 0) PC = BTA
WB	rt = Data		rd = result	

A more complex trace:

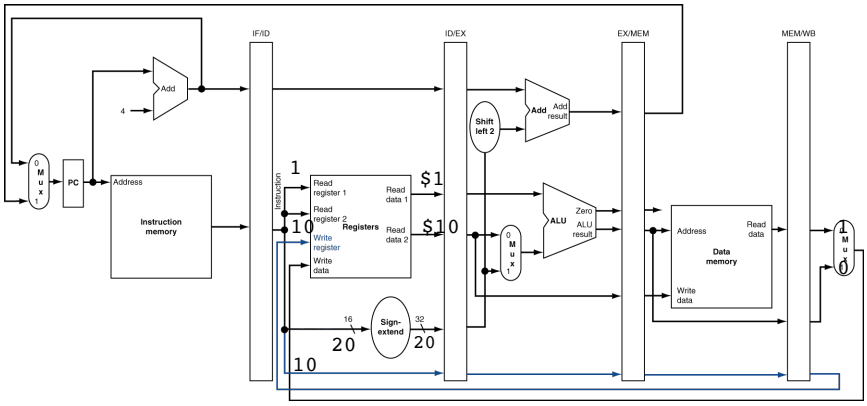
```
lw $10, 20($1)
sub $11, $2, $3
add $12, $3, $4
lw $13, 24($1)
add $14, $5, $6
```


Trace: Cycle 1



lw $10, 20($1)

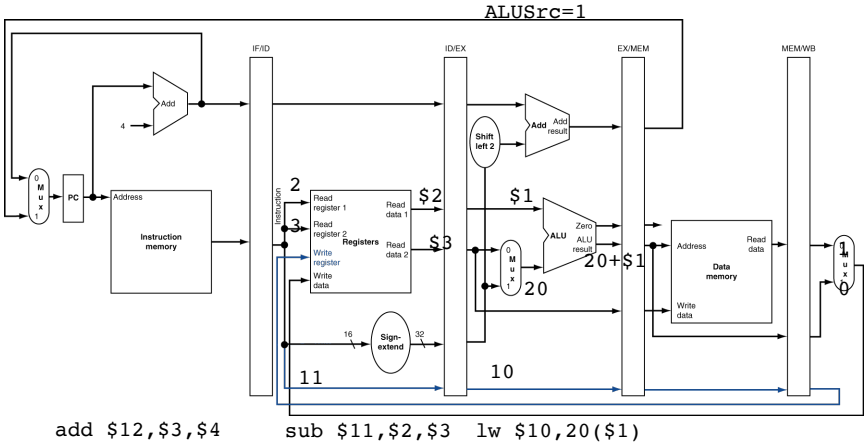
Trace: Cycle 2




```
sub $11, $2, $3      lw $10, 20($1)
```

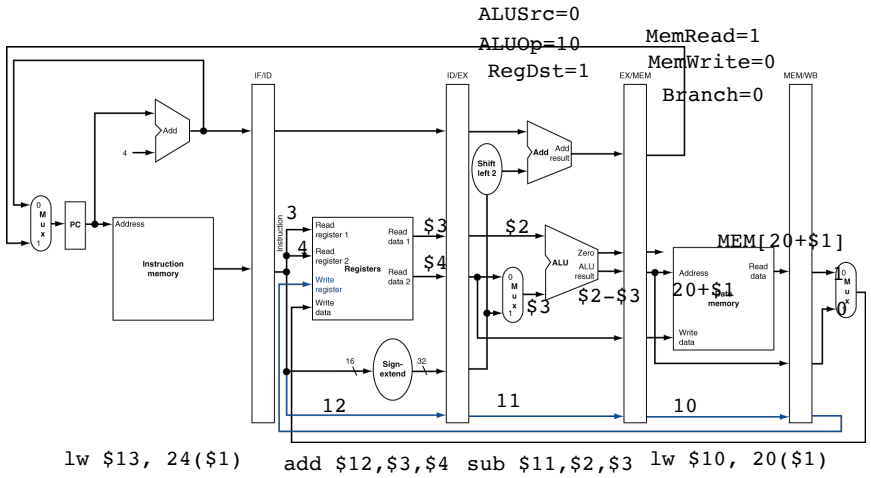

Trace: Cycle 3

```
RegDst=0
ALUOp=00
ALUSrc=1
```

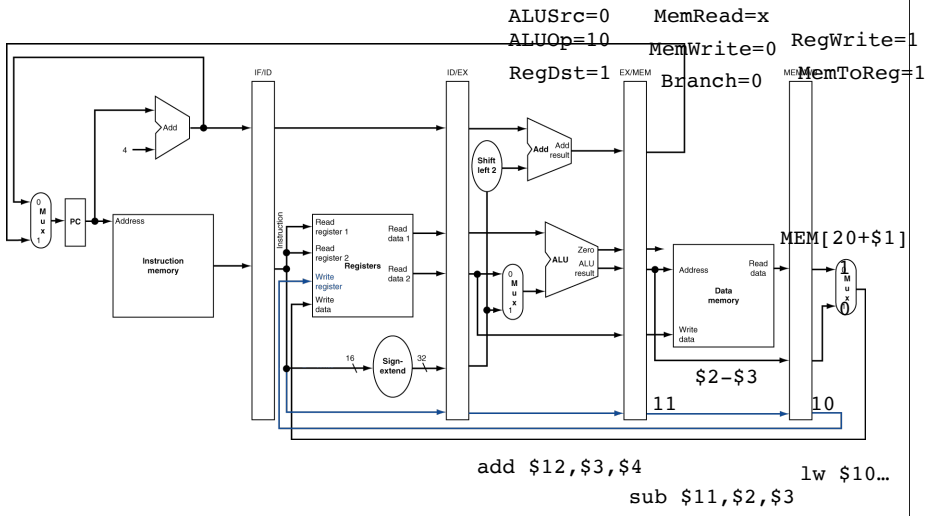



```
add $12,$3,$4      sub $11,$2,$3   lw $10,20($1)
```

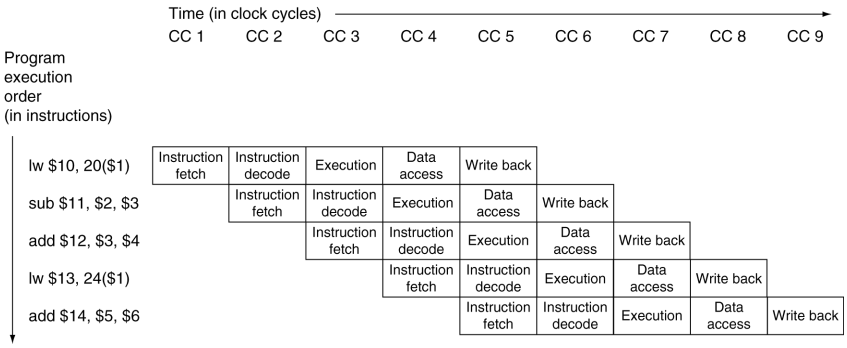
Trace: Cycle 4



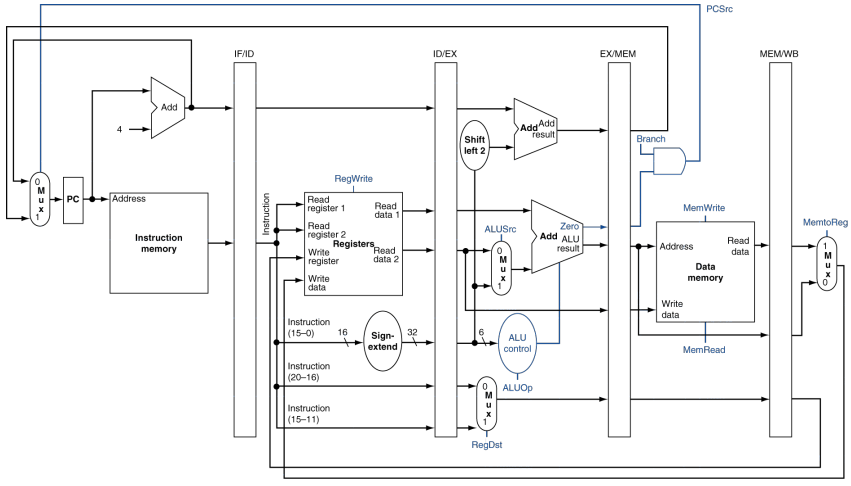
Trace: Cycle 5



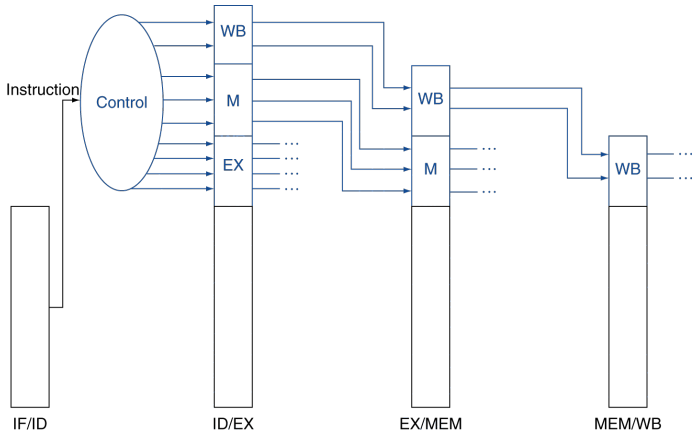
Another view of trace:



Simplified view of pipeline control (Fig. 4.46 p. 301):



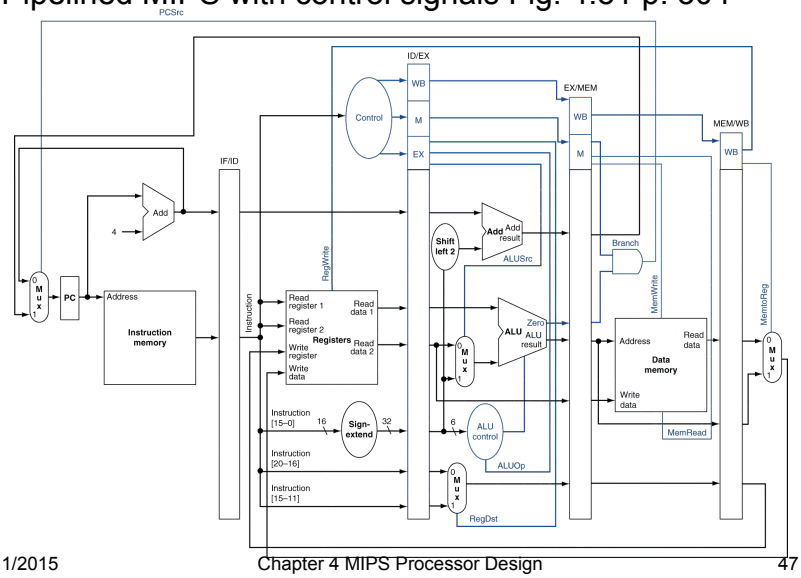
Carrying information between stages (Fig. 4.50 p. 303):



Control signals organized into stages: See Fig. 4.49

	EX	EX	EX	EX	MEM	MEM	MEM	WB	WB
Instr	RegDst	ALUOp1	~~ALUOp2~~ ALUOp0	ALUSrc	Branch	Mem Read	Mem Write	Reg Write	Memto Reg
R-type									
lw									
sw									
beq									

Pipelined MIPS with control signals Fig. 4.51 p. 304



Pipeline hazards

Ideally, start/complete new instruction every cycle in pipeline.

But sometimes this may not be possible: *hazards*

Hazards may result in *stalls* or *bubbles* in pipeline (cycles where nothing happens in a stage)

3 types of pipeline hazards:

- Data hazards
- Control hazards
- Structural hazards